REMARKS

At the outset, the undersigned wishes to thank Exr. Puente for his time and courtesy during the recent telephone interview of March 3, 2008.

As discussed during the interview, the preambles of the independent claims have been amended to recite "**without the use of a test script**," for reasons made clear below. Also, the claims have been amended to further characterize the basis upon which a standard input is selected from the knowledge base of standard inputs; namely, the categorized input field type of the detected user input field.

This Amendment is responsive to the Office Action mailed November 26, 2007, and is filed together with the fee for a one-month extension of time.

Claims 1-36 were rejected as being indefinite, under §112(2). Reconsideration and withdrawal of these rejections are respectfully requested.

The Examiner's kind suggestions to amend the claims have been adopted. The Examiner's attention to detail in this regard is appreciated.

Claims 1-7, 9-18, 20-29 and 31-36 were rejected as being anticipated over Duggan et al. Reconsideration and withdrawal of these rejections are respectfully requested, for the following reasons.

Duggan et al. is squarely representative of the type of testing discussed in the Background section of the present application. Indeed, Duggan et al. rely entirely on pre-packaging scripts, running the scripts and recording the result. Indeed, Duggan et al. are fully representative of the conventional QA tools discussed in the present Background section:

> **[0003] Conventional QA tools for websites and web applications are typically based on a record and playback model, whereby a tester manually logs onto a selected website or web application and runs through a typically transaction or session. The tester's inputs and the path the tester takes through the site are recorded in the form of a script, which may later be played back. The recorded script may be selectively altered, to enable the QA tool to run through a wide variety of user-website interactions. This manual approach is both time consuming and inherently incomplete, as the QA tool relies upon a human created script to guide its progress through the site. Indeed, the tester must key a great many permutations of the inputted data to attempt to fully test the site or application. It is, therefore, likely that conventional QA tools could miss latent errors in the website or web application that could degrade the functionality of the site or application when with certain inputs are entered.**
>
> **[0004] Moreover, current websites and applications are not static constructs, but are frequently upgraded and changed. When the website or application is changed or upgraded, conventional QA tools require that a new script be recorded and the entire process started over again. This record-playback model, therefore, renders conventional tools rather inflexible to even relatively minor changes in the websites and applications for which they are designed.**

Indeed,, Duggan et al. test web pages and web applications by first drafting a script of commands tailored specifically to the web site or web application to be tested. In Duggan et al.'s own words, at Col. 7, lines 30-39:

> Test scripts are ASCII text files containing a list of the names of commands to be executed. Each line of the file contains one command name, and the commands are executed in the order listed. Examples of typical commands include LOGON, PASSWORD, GET_ALL, READ_EM, READ_FAX, READ_VOC, WAIT_SSS, WAIT_SSS_RRR, and LOGOFF. Preferably, all commands are in upper-case type. A test script may have a header starting with the word TEST, as well as lines of comments denoted with exclamation points ("!").

Each time a new web site or web application is to be tested, a new script must be prepared. Duggan et al. teaches an elaborate User Interface to facilitate the authoring of such scripts (see, e.g., Duggan et al.'s Fig. 2). However, a script must be prepared before the web site or web application can be tested, as such script contains a listing of the commands that are to be executed. Col. 12, lines 15-50 detail the steps for creating or modifying a script for a given application program:

FIG. **7** shows the content and arrangement of a Create Script Window **262** that a test operator can use to generate a test script. Buttons **266, 268**, and **270** can be used to create a new test script, open an existing test script, and save any modification made to either a new or existing test script. The file name of the test script being created or modified is shown in field **264** of the window **262**. A list box **272** contains a list of all of the commands in the command module created for testing a given application program. An information box **273** can be used to display a brief description of the highlighted command. A test script can contain any combination of the available commands. A test operator can add a command to a test script by highlighting that command in the command window **272** and then clicking on button **274**. This appends the highlighted command to the test script listing shown in the test script window **278**. In the example shown, the test script being created has four commands—LOGON, WAIT_005, PASSWORD, and WAIT_005. The LOGON command could, for example, cause the computer running the test tool to connect to the server computer on which the application under test is deployed and to initiate access to the application program. The WAIT_005 command could then cause the computer to pause five seconds, simulating, for example, the time it takes a real user to read a logon screen of the application program under test. The PASSWORD command could then cause the computer to issue a username/password combination in order to gain access to the functions of the application program, followed by another WAIT_005 command, and so on. In the present embodiment, a test script can have up to fifty consecutive commands, in any order. A test operator can move up and down in the test script using the Up and Down buttons shown at **280**, and can remove a particular command from the test script by highlighting its name in the test script window **278** and then clicking on the Remove button.

Therefore, it is clear that Duggan envisages applying a predetermined list of commands to a web application and examining the output of the web application resulting from the application of the predetermined command list to the web application.

In contrast, embodiments of the present invention call for "a computer to automatically test a website or a web application", as set out in the preamble, and for the <u>code</u> that generated the page to be examined (automatically, by the computer). Duggan et al. do not do this. A user input field is then detected <u>in the examined code</u>. Duggan et al. do not detect any user input fields in the examined code. Thereafter, a knowledge base of standard inputs is consulted and a standard input is then selected according to the categorized input field type of the detected user input field in the examined code. That is, the standard input is selected based upon a detected user input field in the examined code, which detected user input field is categorized using a knowledge base. In contrast, in Duggan et al., the commands are pre-selected by the pre-existing script, and such commands determine the type of user input fields: a LOGON command in the script will input a suitable LOGON entry, and a password will input a suitable entry of the PASSWORD type – even through the script may randomly choose from among a large number of username/password combinations, as set forth at Col. 7, lines 40-48:

> For application programs that require a user name and password in order to "logon" to the application, the test tool program supports user lists containing from 1 to 10,000 username/password combinations. A user list file is an ascii text file, each line of which contains one username/password combination. User list files have a header starting with the word USER and containing any number of comment lines denoted with exclamation points ("!").

In contrast, embodiments of the present invention do not utilize authored scripts but rely, instead, upon the automatic testing of a website or web application by means of examining the code, detecting user input fields, categorizing the detected user input field and selecting a standard input according to the categorized input field type of the detected user input field in the examined code. That is, embodiments of the present invention do not include a pre-prepared list of commands to execute (i.e., a script), but rely upon the programmatic examination of the underlying code that generated the website or web application and based thereon, selecting a suitable standard input and applying the selected standard input to the detect input field. That is, embodiments of the present inventions, in effect, have no idea what to input to the website or web application before the underlying code thereof has been examined – in direct contrast to the script approach of Duggan et al., as discussed in the Background section of the present application.

To further emphasize the foregoing, claim 1 has been amended as follows:

> **examining a code that generated the page;**
>
> **detecting a user input field in the examined code and categorizing the detected input field according to a type of the input field;**
>
> **consulting a knowledge base of standard inputs, the knowledge base of standard inputs storing a plurality of standard inputs that are categorized according to one of a plurality of input field types;**
>
> **selecting a standard input from the knowledge base of standard inputs, the selected standard input being chosen from among the plurality of standard inputs of the selected input field type according to the categorized input field type of the detected user input field in the examined code and applying the selected standard input to the detected input field;**

That is, the standard input is selected according to the categorized type of the detected input field, which is a step that is clearly not carried out by Duggan et al., whose commands and, therefore, user inputs (as well as the type thereof) are predetermined in advance of testing the website or web application.

Claims 34, 35 and 36 have been similarly amended, as follows:

> **detecting a user input field in the opened page;**
>
> **based upon a type of the detected user input field in the opened page, selecting a standard input from a knowledge base of standard inputs that stores a plurality of standard inputs, and applying the selected standard input to the detected input field;**

Duggan et al., in contrast, does not examine the underlying code, detect any user input fields in the opened page or select a standard input based upon a type of the detected user input field in the opened page, as positively recited herein. Therefore, the claims cannot be anticipated by Duggan et al. Reconsideration and withdrawal of the 35 USC §102(b) rejections applied to the claims are, therefore, respectfully requested.

Applicant's attorney, therefore, respectfully submits the present application is in condition for an early allowance and passage to issue. If any unresolved issues remain, please contact the undersigned attorney of record at the telephone number indicated below and whatever is needed will be done immediately.

Respectfully submitted,

[illegible]

Date: March 4, 2008 By:

Alan W. Young
Attorney for Applicant
Registration No. 37,970

YOUNG LAW FIRM, P.C.
4370 Alpine Road, Suite 106
Portola Valley, CA 94028
Tel.: (650) 851-7210
Fax: (650) 851-7232

\\Ylfserver\ylf\CLIENTS\ORCL\5863 (OID-2003-070-01)\5863 AMEND.1.doc